HW 2 | APPM 4650 | Soroush Khadem

1.) a.) $\lim_{x\to\infty}\left|\frac{e^x}{x}\right| \underset{\text{L'Hop}}{=} \lim_{x\to\infty}\frac{e^x}{1} = \infty$, so $e^x \neq O(x)$

b.) To show $f=\Theta(g)$, show $f=O(g)$ & $g=O(f)$

$$\lim_{x\to 0}\left|\frac{x\sin(\sqrt{x})}{x^{3/2}}\right| = \lim_{x\to 0}\left|\frac{\sin(\sqrt{x})}{\sqrt{x}}\right| \overset{\text{small angle}}{=} \lim_{x\to 0}\left|\frac{\sqrt{x}}{\sqrt{x}}\right| = 1 < \infty \checkmark$$

$$\lim_{x\to 0}\left|\frac{x^{3/2}}{x\sin(\sqrt{x})}\right| = \lim_{x\to 0}\left|\frac{\sqrt{x}}{\sin(\sqrt{x})}\right| \underset{\text{Small angle}}{=} \lim_{x\to 0}\left|\frac{\sqrt{x}}{\sqrt{x}}\right| = 1 < \infty \checkmark$$

So, $x\sin\sqrt{x} = \Theta(x^{3/2})$

c.) To show $f=o(g)$, $\lim_{x\to\infty}\frac{f}{g}=0$:

$$\lim_{t\to\infty}\frac{e^{-t}}{1/t^2} = \lim_{t\to\infty} t^2 e^{-t} = \lim_{t\to\infty}\frac{t^2}{e^t} \overset{\text{L'Hop}}{=} \lim_{t\to\infty}\frac{2t}{e^t} = \lim_{t\to\infty}\frac{2}{e^t} = 0 \checkmark$$

d.) $\int_0^\varepsilon e^{-x^2}dx = O(\varepsilon)$, $\varepsilon\to 0$. Show that $\lim_{\varepsilon\to 0}\left|\frac{\int_0^\varepsilon e^{-x^2}dx}{\varepsilon}\right| < \infty$

$$\lim_{\varepsilon\to 0}\underbrace{\frac{1}{\varepsilon}\int_0^\varepsilon e^{-x^2}dx}_{\substack{\text{By MVT, } \exists c\in[0,\varepsilon] \text{ s.t.} \\ \text{this } = f(c)}} = \lim_{\varepsilon\to 0} e^{-c^2} \underset{\substack{\text{Since } c\in[0,\varepsilon], \\ \text{as } \varepsilon\to 0,\ c\to 0}}{=} e^0 = 1 < \infty \checkmark$$

e.) $\lim_{x\to 0}\frac{-x/\log(x)}{x} = \frac{-1}{\log(x)} = 0$, so $\frac{-x}{\log(x)} = o(x)$

$$\lim_{x\to 0}\frac{-x/\log(x)}{x^2} = \frac{-1}{x\log(x)} = \frac{-1/x}{\log(x)} \underset{\text{L'Hop}}{=} \frac{1}{x^2}\Big/\frac{1}{x}\ \ -\frac{1}{x} = \infty$$

So $\frac{-x}{\log(x)} \neq O(x^2)$

hw2

September 11, 2020

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        plt.style.use('seaborn-bright')
        plt.rcParams['figure.figsize'] = [15, 5]
```

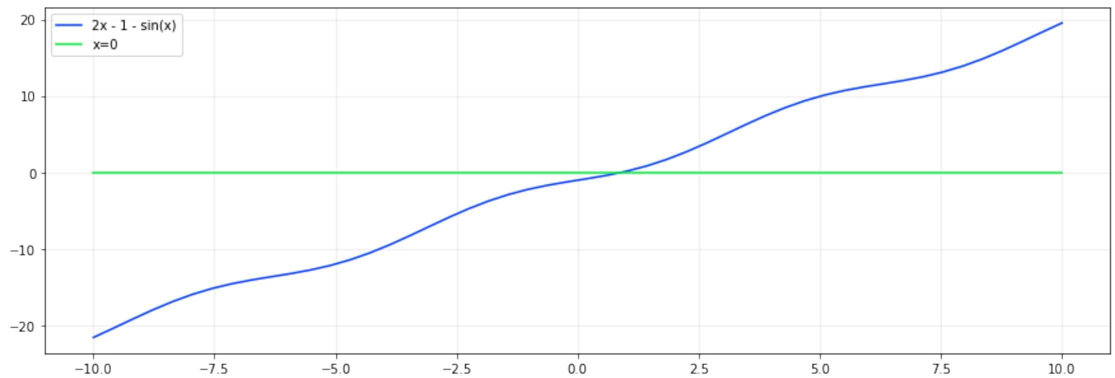
1 Problem 2

```
In [2]: def bisection(func, interval, tol, max_iter = 100):
            low = interval[0]
            high = interval[1]
            if np.sign(func(low)) == np.sign(func(high)):
                raise ValueError("No root in interval [%s, %s]"%(low, high))

            midpoint = np.mean([low, high])
            history = []
            while ((high - low) / 2) >= tol and len(history) < max_iter:
                # split the interval
                midpoint = np.mean([low, high])

                if func(midpoint) == 0:
                    break

                # Figure out which side to go to
                if np.sign(func(midpoint)) != np.sign(func(low)):
                    high = midpoint
                else:
                    low = midpoint

                history.append(midpoint)

            return midpoint, history
```

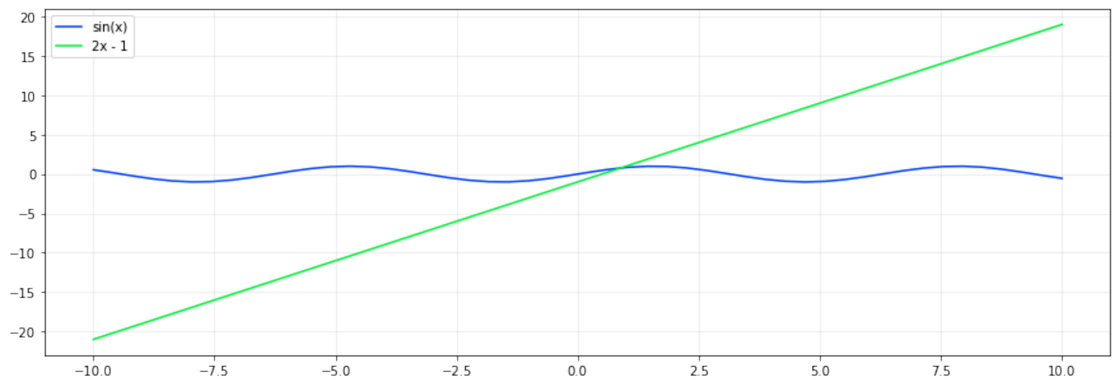
2 Problem 3c

```
In [3]: f = lambda x : 2*x - 1 - np.sin(x)
        x = np.linspace(-10, 10)
```

```
plt.plot(x, f(x))
plt.grid(alpha=0.25)
plt.plot(x, 0*x)
_ = plt.legend(["2x - 1 - sin(x)", "x=0"])
plt.show()
```



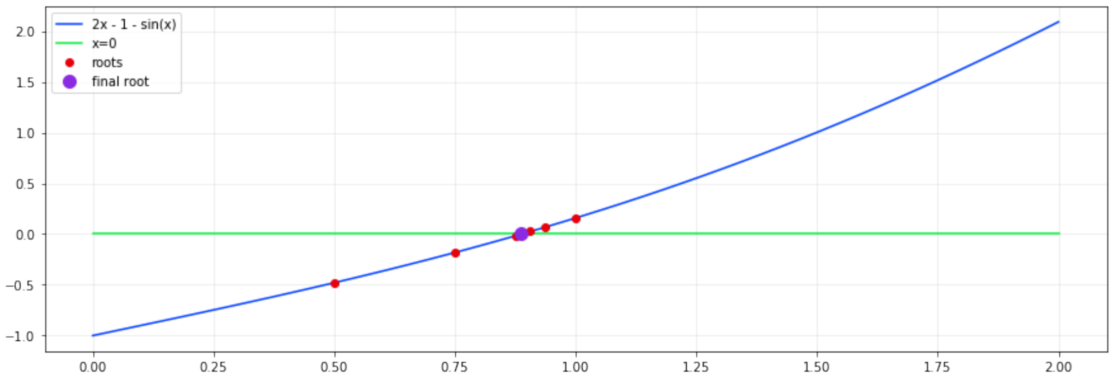

```
In [4]: x = np.linspace(-10, 10)
        plt.plot(x, np.sin(x))
        plt.plot(x, 2*x - 1)
        plt.grid(alpha=0.25)
        _ = plt.legend(["sin(x)", "2x - 1"])
        plt.show()
```




```
In [5]: root, history = bisection(f, [0,2], tol = 1e-8)

        for i, root in enumerate(history):
            print("Iter {:2d}, r={:.16f}".format(i,root))
```

```
Iter  0, r=1.0000000000000000
Iter  1, r=0.5000000000000000
Iter  2, r=0.7500000000000000
Iter  3, r=0.8750000000000000
Iter  4, r=0.9375000000000000
Iter  5, r=0.9062500000000000
Iter  6, r=0.8906250000000000
Iter  7, r=0.8828125000000000
Iter  8, r=0.8867187500000000
Iter  9, r=0.8886718750000000
Iter 10, r=0.8876953125000000
Iter 11, r=0.8881835937500000
Iter 12, r=0.8879394531250000
Iter 13, r=0.8878173828125000
Iter 14, r=0.8878784179687500
Iter 15, r=0.8878479003906250
Iter 16, r=0.8878631591796875
Iter 17, r=0.8878555297851562
Iter 18, r=0.8878593444824219
Iter 19, r=0.8878612518310547
Iter 20, r=0.8878622055053711
Iter 21, r=0.8878626823425293
Iter 22, r=0.8878624439239502
Iter 23, r=0.8878623247146606
Iter 24, r=0.8878622651100159
Iter 25, r=0.8878622353076935
Iter 26, r=0.8878622204065323
```

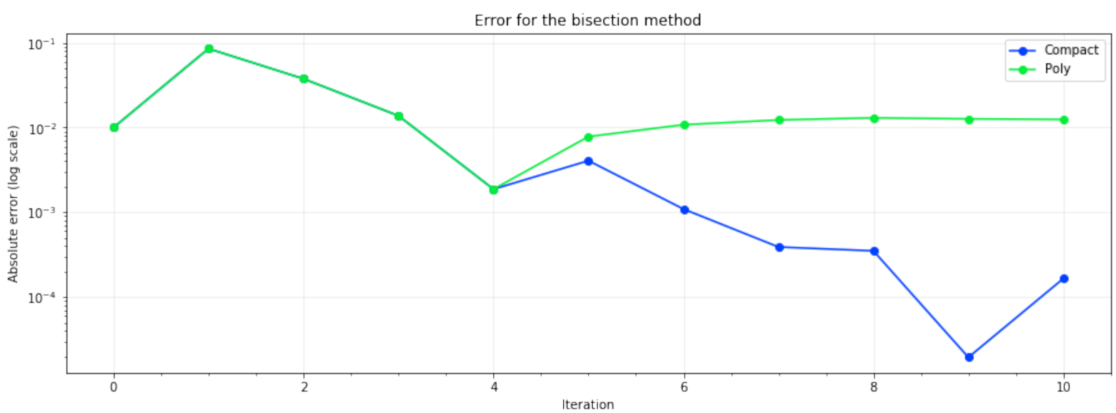
In [17]:
```
x = np.linspace(0,2)
plt.plot(x, f(x))
plt.grid(alpha=0.25)
plt.plot(x, 0*x)
plt.plot(history, f(np.array(history)), 'o')
plt.plot(root, f(root), 'o', markersize=10)
_ = plt.legend(["2x - 1 - sin(x)", "x=0", "roots", "final root"])
plt.show()
```



3 Probelm 3d

In [7]:
```
compact = lambda x : (x-5)**9
poly = lambda x : np.polyval(np.poly([5]*9), x)
root_compact, history_compact = bisection(compact, (4.82, 5.2), tol=1e-4, max_iter=20)
root_poly, history_poly = bisection(poly, (4.82, 5.2), tol=1e-4, max_iter=20)
```

In [8]:
```
# Plot absolute error
true_answer = 5
plt.plot(abs(np.array(history_compact) - true_answer) ,'o-')
plt.plot(abs(np.array(history_poly) - true_answer) ,'o-')
plt.yscale('log')
plt.grid(alpha=0.25)
plt.minorticks_on()
plt.ylabel("Absolute error (log scale)");
plt.xlabel("Iteration");
plt.title("Error for the bisection method");
_ = plt.legend(["Compact", "Poly"])
plt.show()
```



In [9]:
```
# Run with lower tolerance to get a more complete plot
root_compact, history_compact = bisection(compact, (4.82, 5.2), tol=1e-16, max_iter=20
root_poly, history_poly = bisection(poly, (4.82, 5.2), tol=1e-16, max_iter=20)
# Plot absolute error
true_answer = 5
fig, ax = plt.subplots()
plt.plot(abs(np.array(history_compact) - true_answer) ,'o-')
plt.plot(abs(np.array(history_poly) - true_answer) ,'o-')
ax.set_yscale('log')
```

3.) $2x-1 = \sin x \Rightarrow 2x-1-\sin x = 0$

a.) Define $g(x) = 2x-1-\sin x$, looking graphically, interval is $[0,2]$.
$[0,2]$ has range $[-1, 3-\sin(2) \approx 2]$, so since $g(x)$ changes sign, a root exists, by the MVT

b.) Looking at the derivative of the function: $2 \underbrace{-\cos x}_{\in[1,-1]}$

$2-\cos(x) > 0 \quad \forall x \in \mathbb{R}$, so it is monotonically increasing.
This means it can only change signs once.

c.) (jupyter)

d.) (jupyter)

4.) $g_\mu(x) = \mu \cdot \min(x, 1-x) = \mu(-|x-1/2| + 1/2)$

a.) Show for $\mu \in [0,2]$, $x \in [0,1]$, $g_\mu(x) \in [0,1]$

For $\mu = 0$, $g_0(x) = 0 \quad \forall x \in [0,1]$

For $\mu \in (0,2]$, g_μ is increasing, so max occurs at $\mu = 2$.

Endpoints: $g_2(0) = 2\min(0, 1-0) = 0$

$g_2(1) = 2\min(1,0) = 0$

Max val: $g_2'(x) = \dfrac{-2(x-1/2)}{|x-1/2|} = 0 \Rightarrow x = 1/2$

$g_2(1/2) = 1$

↳ so, $g_{[0,2]}([0,1]) \in [0,1]$

b.) $0 \le \mu < 1$, $\exists$ unique fixed point in $[0,1]$

Check contraction: $g_\mu'(x) = \frac{-\mu(x-1/2)}{|x-1/2|}$, $|g_\mu'(x)| = \frac{\mu|x-1/2|}{|x-1/2|}$

$= \mu$

$\hookrightarrow$ Since $0 \le \mu < 1$, $\mu < 1$ so there is a fixed point on $[0,1]$. Yes, contraction mapping applies

c.) If $\mu = 1$, then $g_1(x) = -|x-1/2| + 1/2$. On the interval $[0, 1/2]$: $|x-1/2| = -(x-1/2) = -x+1/2$.

So, $g_1(x) = -(-x+1/2)+1/2 = x$. Thus, there are infinite fixed points.

d.) For $1 < \mu \le 2$: $g_\mu = -\mu|x-1/2| + \mu/2$

at $x=0$: $g_\mu(0) = -\mu/2 + \mu/2 = 0$ so this is a fixed point.

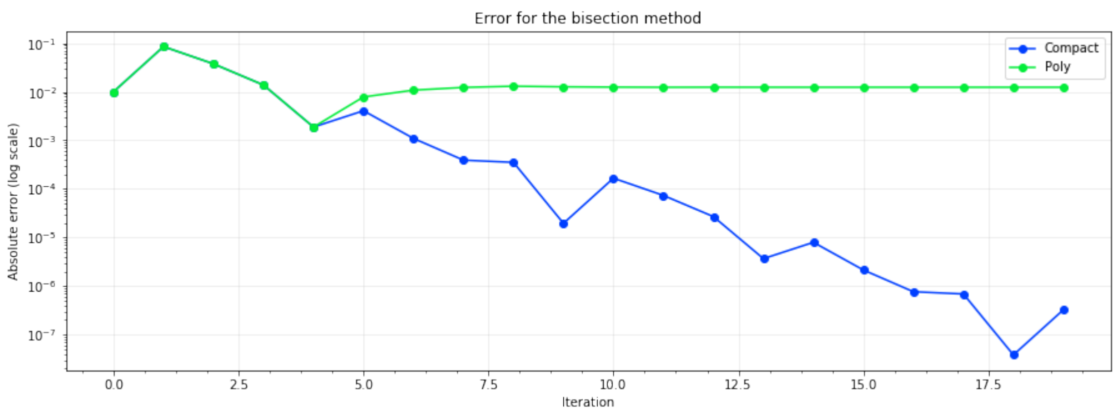
For $x \in (1/2, 1]$, $g_\mu = -\mu(x-1/2) + \mu/2 = -\mu x + \mu = \mu(1-x)$

Fixed point: $x = \mu(1-x) \Rightarrow x + \mu x = \mu \Rightarrow x = \mu/1+\mu$

Thus, there are two fixed points: 0 and $\mu/1+\mu$.

No, contraction does not apply since $g_\mu'(x) = \mu > 1$.

```
ax.grid(alpha=0.25)
ax.minorticks_on()
ax.set_ylabel("Absolute error (log scale)");
ax.set_xlabel("Iteration");
ax.set_title("Error for the bisection method");

_ = plt.legend(["Compact", "Poly"])
plt.show()
```



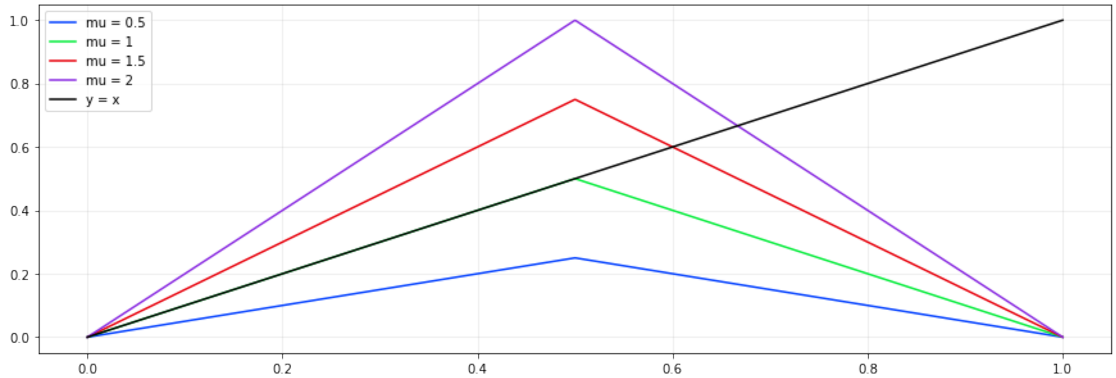
What is going on: It seems that, due to the lack of precision of the fully expanded polynomial version, the accuracy is no longer decreasing, so it has gotten "stuck", and cannot improve the guess.

4 Problem 4

```
In [10]: # Define tent function to work with lists and with single values
         def tent(mu, x):
             if np.shape(x) == ():
                 # single value
                 min_func = min
             else:
                 # list, do elementwise
                 min_func = np.minimum
             return mu * min_func(x, 1-x)
```

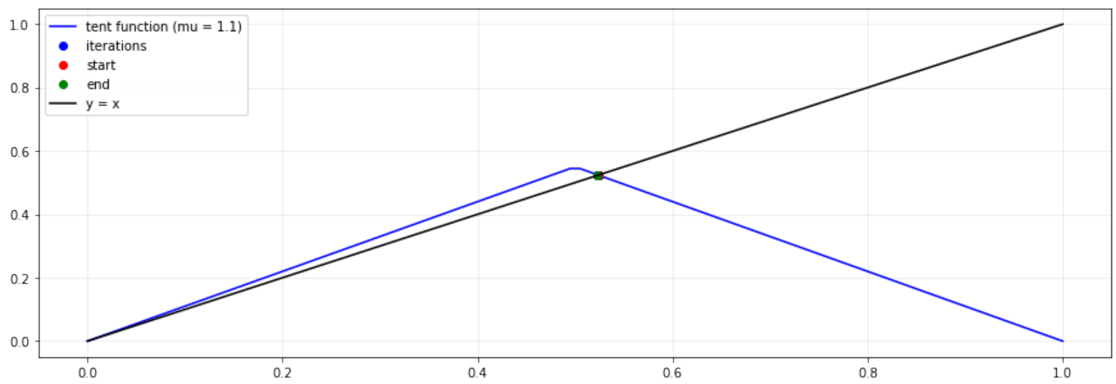
```
In [11]: # Visualize tent for a few values of mu
         x = np.linspace(0,1, 1000)
         plt.plot(x, tent(0.5, x))
         plt.plot(x, tent(1, x))
         plt.plot(x, tent(1.5, x))
         plt.plot(x, tent(2, x))
         plt.plot(x, x, 'k')
         plt.grid(alpha=0.25)
```

```
_ = plt.legend(["mu = 0.5", "mu = 1", "mu = 1.5", "mu = 2", "y = x"])
plt.show()
```

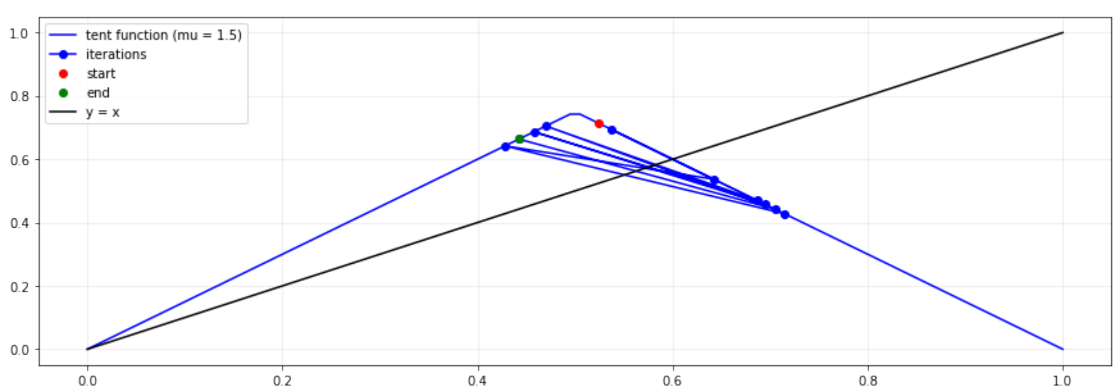


In [12]:
```
# Function to find fixed point of the tent function
def tent_fixed_point(mu, x0, maxiter=100, tol=1e-8):
    error = np.inf
    curr_iter = 0
    xs = []
    while(error > tol and curr_iter < maxiter):
        # iterate
        x = tent(mu, x0)
        error = np.linalg.norm(x0 - x)
        # append to history
        xs.append(x)
        curr_iter += 1
        x0 = x
    return x, xs
```

In [13]:
```
x_start = np.pi/6
mu = 1.1
xf, x_history = tent_fixed_point(mu, x_start, maxiter=10)
x_history = np.array(x_history)
x = np.linspace(0,1,100)
y = tent(mu, x)
plt.plot(x, y, 'b')
plt.plot(x_history, tent(mu, x_history), 'bo')
plt.plot(x_start, tent(mu, x_start), 'ro')
plt.plot(xf, tent(mu, xf), 'go')
plt.plot(x,x,'k')
plt.grid(alpha=0.25)
_ = plt.legend(["tent function (mu = %s)"%mu, "iterations", "start", "end", "y = x"])
plt.show()
```

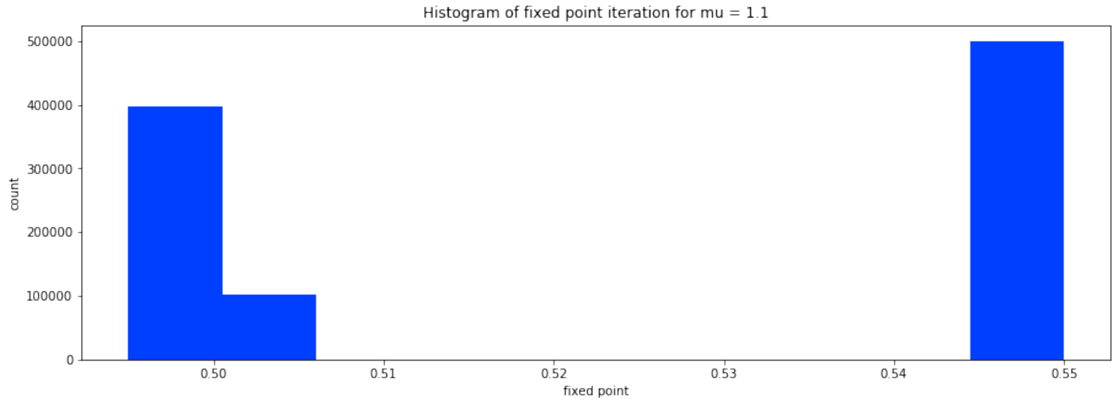


In [14]:
```
x_start = np.pi/6
mu = 1.5
xf, x_history = tent_fixed_point(mu, x_start, maxiter=10)
x_history = np.array(x_history)
x = np.linspace(0,1,100)
y = tent(mu, x)
plt.plot(x, y, 'b')
plt.plot(x_history, tent(mu, x_history), '-bo')
plt.plot(x_start, tent(mu, x_start), 'ro')
plt.plot(xf, tent(mu, xf), 'go')
plt.plot(x,x,'k')
plt.grid(alpha=0.25)
_ = plt.legend(["tent function (mu = %s)"%mu, "iterations", "start", "end", "y = x"])
plt.show()
```



In [15]:
```
x_start = np.pi/6
mu = 1.1
xf, x_history = tent_fixed_point(mu, x_start, maxiter=10**6)
plt.hist(x_history)
```

```
plt.title("Histogram of fixed point iteration for mu = %s"%mu)
plt.xlabel("fixed point")
plt.ylabel("count")
plt.show()
```




```
In [16]: x_start = np.pi/6
         mu = 1.5
         xf, x_history = tent_fixed_point(mu, x_start, maxiter=10**6)
         plt.hist(x_history)
         plt.title("Histogram of fixed point iteration for mu = %s"%mu)
         plt.xlabel("fixed point")
         plt.ylabel("count")
         plt.show()
```